Terms used
**processing sequence pipeline non pipeline snoop coherency**

Found **129** of **157,956**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 129 Result page: **1** 2 3 4 5 6 7 next

Relevance scale

**1** An evaluation of directory schemes for cache coherence

A. Agarwal, R. Simoni, J. Hennessy, M. Horowitz

May 1988 **ACM SIGARCH Computer Architecture News , Proceedings of the 15th Annual International Symposium on Computer architecture**, Volume 16 Issue 2

Full text available: pdf(1.35 MB) Additional Information: full citation, abstract, references, citings, index terms

The problem of cache coherence in shared-memory multiprocessors has been addressed using two basic approaches: directory schemes and snoopy cache schemes. Directory schemes have been given less attention in the past several years, while snoopy cache methods have become extremely popular. Directory schemes for cache coherence are potentially attractive in large multiprocessor systems that are beyond the scaling limits of the snoopy cache schemes. Slight modifications to directory schemes can ...

**2** An evaluation of directory schemes for cache coherence

Anant Agarwal, Richard Simoni, John Hennessy, Mark Horowitz

August 1998 **25 years of the international symposia on Computer architecture (selected papers)**

Full text available: pdf(1.31 MB) Additional Information: full citation, references, index terms

**3** Measuring memory hierarchy performance of cache-coherent multiprocessors using micro benchmarks

Cristina Hristea, Daniel Lenoski, John Keen

November 1997 **Proceedings of the 1997 ACM/IEEE conference on Supercomputing (CDROM)**

Full text available: pdf(97.47 KB) Additional Information: full citation, abstract, references, citings

Even with today's large caches, the increasing performance gap between processors and memory systems imposes a memory bottleneck for many important scientific and commercial applications. This bottleneck is intensified in shared-memory multiprocessors by contention and the effects of cache coherency. Under heavy memory contention, the memory latency may increase 2 or 3 times. Nonethless, as more sophisticated techniques are used to hide latency and increase bandwidth, measuring memory performanc ...

**4** Memory access buffering in multiprocessors

M. Dubois, C. Scheurich, F. Briggs
June 1986 **ACM SIGARCH Computer Architecture News , Proceedings of the 13th annual international symposium on Computer architecture**, Volume 14 Issue 2

Full text available: pdf(943.66 KB) Additional Information: full citation, abstract, references, citings, index terms

In highly-pipelined machines, instructions and data are prefetched and buffered in both the processor and the cache. This is done to reduce the average memory access latency and to take advantage of memory interleaving. Lock-up free caches are designed to avoid processor blocking on a cache miss. Write buffers are often included in a pipelined machine to avoid processor waiting on writes. In a shared memory multiprocessor, there are more advantages in buffering memory requests, since each m ...

5 Pipeline Architecture
C. V. Ramamoorthy, H. F. Li
January 1977 **ACM Computing Surveys (CSUR)**, Volume 9 Issue 1

Full text available: pdf(3.53 MB) Additional Information: full citation, references, citings, index terms

6 Cache: Effective stream-based and execution-based data prefetching
Sorin Iacobovici, Lawrence Spracklen, Sudarshan Kadambi, Yuan Chou, Santosh G. Abraham
June 2004 **Proceedings of the 18th annual international conference on Supercomputing**

Full text available: pdf(1.35 MB) Additional Information: full citation, abstract, references, index terms

With processor speeds continuing to outpace the memory subsystem, cache missing memory operations continue to become increasingly important to application performance. In response to this continuing trend, most modern processors now support hardware (HW) prefetchers, which act to reduce the missing loads observed by an application.This paper analyzes the behavior of cache-missing loads in SPEC CPU2000 and highlights the inability of unit and single non-unit stride prefetchers to correctly prefet ...

**Keywords**: hardware prefetcher, multiple strides, stream prefetching

7 Software pipelining loops with conditional branches
Mark G. Stoodley, Corinna G. Lee
December 1996 **Proceedings of the 29th annual ACM/IEEE international symposium on Microarchitecture**

Full text available: pdf(1.64 MB) Additional Information: full citation, abstract, references, citings, index terms

Software pipelining is an aggressive scheduling technique that generates efficient code for loops and is particularly effective for VLIW architectures. Few software pipelining algorithms, however, are able to efficiently schedule loops that contain conditional branches. We have developed an algorithm we call All Paths Pipelining (APP) that addresses this shortcoming of software pipelining. APP is designed to achieve optimal or near-optimal performance for any run of iterations while providing ef ...

8 Automatic transformation of series expressions into loops
Richard C. Waters
January 1991 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 13 Issue 1

Full text available: pdf(3.36 MB) Additional Information: full citation, abstract, references, citings, index terms, review

The benefits of programming in a functional style are well known. In particular, algorithms

that are expressed as compositions of functions operating on sequences/vectors/streams of data elements are easier to understand and modify than equivalent algorithms expressed as loops. Unfortunately, this kind of expression is not used anywhere near as often as it could be, for at least three reasons: (1) most programmers are less familiar with this kind of expression than with loops; (2) most pro ...

**Keywords**: sequences, series, streams, vectors

**9** Pipelined data parallel algorithms—concept and modeling
C.-T. King, W.-H. Chou, L. M. Ni
June 1988 **Proceedings of the 2nd international conference on Supercomputing**

Full text available: pdf(1.22 MB) Additional Information: full citation, abstract, references, citings, index terms

A new style of efficient parallel algorithms on distributed-memory multiprocessors is introduced, which exploits parallelism through pipelined parallel computation, or large-grain pipelining. By using macro-pipelining between nodes in the system, large-grain pipelining regulates the flows of data in the multiprocessor so that the degree of overlapping can be maximized and the effect of communication overhead can be minimized. To model pipelined parallel computations, an ana ...

**10** A compilation technique for software pipelining of loops with conditional jumps
Kemal Ebcioğlu
December 1987 **Proceedings of the 20th annual workshop on Microprogramming**

Full text available: pdf(1.48 MB) Additional Information: full citation, abstract, references, citings

We describe a compilation algorithm for efficient software pipelining of general inner loops, where the number of iterations and the time taken by each iteration may be unpredictable, due to arbitrary if-then- else statements and conditional exit statements within the loop. As our target machine, we assume a wide instruction word architecture that allows multi-way branching in the form of if-then-else trees, and that allows conditional register transfers depending on where the microinstruct ...

**11** Towards efficient fine-grain software pipelining
Guang R. Gao, Herbert H. J. Hum, Yue-Bong Wong
June 1990 **ACM SIGARCH Computer Architecture News , Proceedings of the 4th international conference on Supercomputing**, Volume 18 Issue 3

Full text available: pdf(1.15 MB) Additional Information: full citation, abstract, references, index terms

Dataflow software pipelining was proposed as a means of structuring fine-grain parallelism and has been studied mostly under an idealized dataflow architecture model with infinite resources[9]. In this paper, we investigate the effects of software pipelining under realistic architecture models with finite resources. Our target architecture is the McGill Dataflow Architecture which employs conventional pipelined techniques to achieve fast instruction executi ...

**12** SIMP (Single Instruction stream/Multiple instruction Pipelining): a novel high-speed single-processor architecture
K. Murakami, N. Irie, S. Tomita
April 1989 **ACM SIGARCH Computer Architecture News , Proceedings of the 16th annual international symposium on Computer architecture**, Volume 17 Issue 3

Full text available: pdf(1.23 MB) Additional Information: full citation, abstract, references, citings, index terms

SIMP is a novel multiple instruction-pipeline parallel architecture. It is targeted for enhancing the performance of SISD processors drastically by exploiting both temporal and

spatial parallelisms, and for keeping program compatibility as well. Degree of performance enhancement achieved by SIMP depends on; i) how to supply multiple instructions continuously, and ii) how to resolve data and control dependencies effectively. We have devised the outstanding techniques for instruction fetch an ...

**13** IP Design and Reuse: Synthesis of pipelined memory access controllers for streamed data applications on FPGA-based computing engines

Joonseok Park, Pedro C. Diniz

September 2001 **Proceedings of the 14th international symposium on Systems synthesis**

Full text available: pdf(193.12 KB) Additional Information: full citation, abstract, references, citings, index terms

Commercially available behavioral synthesis tools do not adequately support FPGA vendor-specific external memory interfaces making it extremely difficult to exploit pipelined memory access modes as well as application specific memory operations scheduling critical for high-performance solutions. This lack of support substantially increases the complexity and the burden on designers in the mapping of applications to FPGA-based computing engines. In this paper we address the problem of external me ...

**Keywords**: FPGA-based configurable computing, hardware interfaces and customizable memory controllers, scheduling of memory accesses

**14** A bandwidth-efficient architecture for media processing

Scott Rixner, William J. Dally, Ujval J. Kapasi, Brucek Khailany, Abelardo López-Lagunas, Peter R. Mattson, John D. Owens

November 1998 **Proceedings of the 31st annual ACM/IEEE international symposium on Microarchitecture**

Full text available: pdf(1.32 MB) Additional Information: full citation, references, citings, index terms

**15** A 32-bit CMOS microprocessor with six-stage pipeline structure

H. Kaneko, Y. Miki, S. Nohara, K. Koya, M. Araki

November 1986 **Proceedings of 1986 ACM Fall joint computer conference**

Full text available: pdf(831.34 KB) Additional Information: full citation, references, citings, index terms

**16** Software for Reconfigurable Systems: Performance-constrained pipelining of software loops onto reconfigurable hardware

Greg Snider

February 2002 **Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays**

Full text available: pdf(206.27 KB) Additional Information: full citation, abstract, references, citings

*Retiming and slowdown* are algorithms that can be used to pipeline synchronous circuits. *Iterative modulo scheduling* is an algorithm for software pipelining in the presence of resource constraints. Integrating the best features of both yields a pipelining algorithm, *retimed modulo scheduling*, that can more effectively exploit the idiosyncrasies of reconfigurable hardware. It also fits naturally into a design space exploration process to trade-off speed for power, energy or ar ...

**17** Networks III: A near optimal scheduler for switch-memory-switch routers

Adnan Aziz, Amit Prakash, Vijaya Ramachandra

June 2003 **Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures**

Full text available: pdf(1.10 MB) Additional Information: full citation, abstract, references, citings, index terms

We present a simple and near optimal randomized parallel scheduling algorithm for scheduling packets in routers based on the *Switch-Memory-Switch* (*SMS*)architecture, which emulates 'output queuing' by using a collection of small memories within the switch to buffer packets, and which forms the basis of the fastest routers in use today. For a router with $N$ inputs and $N$ outputs, our algorithm computes the schedule in $O(\log^* N)$ rounds, where a round is a com ...

**Keywords**: matching, parallelism, randomization, routers, schedulers

**18** Computational power of pipelined memory hierarchies

Gianfranco Bilardi, Kattamuri Ekanadham, Pratap Pattnaik

July 2001 **Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures**

Full text available: pdf(301.56 KB) Additional Information: full citation, abstract, references, citings, index terms

We define a model of computation, called the *Pipelined Hierarchical Random Access Machine* with *access function a* ($x$), denoted the $a(x)$-PH-RAM. In this model, a processor interacts with a memory which can accept requests at a constant rate and satisfy each of the requests to the location $x$ within $a(x)$ units of time.

We investigate memory management strategies that lead to time efficient implementations of arbitrary computations on a PH-R ...

**19** System design methodologies and experiences: Artificial neural network implementation on a single FPGA of a pipelined on-line backpropagation

Rafael Gadea, Joaquín Cerdá, Franciso Ballester, Antonio Mocholí

September 2000 **Proceedings of the 13th international symposium on System synthesis**

Full text available: pdf(179.87 KB) Additional Information: full citation, abstract, references

The paper describes the implementation of a systolic array for a multilayer perceptron on a Virtex XCV400 FPGA with a hardware-friendly learning algorithm. A pipelined adaptation of the on-line backpropagation algorithm is shown. Parallelism is better exploited because both forward and backward phases can be performed simultaneously. We can implement very large interconnection layers by using large Xilinx devices with embedded memories alongside the projection used in the systolic architecture. ...

**20** The optimum pipeline depth considering both power and performance

A. Hartstein, Thomas R. Puzak

December 2004 **ACM Transactions on Architecture and Code Optimization (TACO)**, Volume 1 Issue 4

Full text available: pdf(351.52 KB) Additional Information: full citation, abstract, references, index terms

The impact of pipeline length on both the power and performance of a microprocessor is explored both by theory and by simulation. A theory is presented for a range of power/performance metrics, $BIPS^m/W$. The theory shows that the more important power is to the metric, the shorter the optimum pipeline length that results. For typical parameters neither $BIPS/W$ nor $BIPS^2/W$ yield an optimum, i.e., a non-pipelined design is optimal. For $BIPS^3/W$< ...

***Keywords***: *Pipeline Depth, Power and Performance, Simulation, Workload Specificity*

Results 1 - 20 of 129 Result page: **1** 2 3 4 5 6 7 next

| Ref # | Hits | Search Query | DBs | Default Operator | Plurals | Time Stamp |
|---|---|---|---|---|---|---|
| L1 | 42 | (Michael near Mayfield).in. | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | OR | OFF | 2005/06/24 15:55 |
| L2 | 158 | (prevent$4 or avoid$4) near3 livelock$2 | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | OR | OFF | 2005/06/24 15:55 |
| L3 | 42 | coheren$4 adj conflict$4 | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | OR | OFF | 2005/06/24 15:55 |
| L4 | 6315 | snoop$4 | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | OR | OFF | 2005/06/24 15:55 |
| L5 | 152860 | pipelin$4 or nonpipelin$4 | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | OR | OFF | 2005/06/24 15:55 |
| L6 | 158 | (prevent$4 or avoid$4) near3 livelock$2 | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | OR | OFF | 2005/06/24 15:55 |
| L7 | 42 | coheren$4 adj conflict$4 | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | OR | OFF | 2005/06/24 15:55 |
| L8 | 1 | L6 and L7 | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | OR | OFF | 2005/06/24 15:55 |
| L9 | 6315 | snoop$4 | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | OR | OFF | 2005/06/24 15:55 |

| | | | | | | |
|---|---|---|---|---|---|---|
| L10 | 53 | L6 and L9 | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | OR | OFF | 2005/06/24 15:55 |
| L11 | 53 | L6 and L9 | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | OR | OFF | 2005/06/24 15:55 |
| L12 | 152860 | pipelin$4 or nonpipelin$4 | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | OR | OFF | 2005/06/24 15:55 |
| L13 | 53 | L6 and L9 | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | OR | OFF | 2005/06/24 15:55 |
| L14 | 32 | L13 and L12 | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | OR | OFF | 2005/06/24 15:55 |
| L15 | 116846 | coheren$4 | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | OR | OFF | 2005/06/24 15:55 |
| L16 | 32 | L13 and L12 | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | OR | OFF | 2005/06/24 15:55 |
| L17 | 116846 | coheren$4 | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | OR | OFF | 2005/06/24 15:55 |
| L18 | 20 | L16 and L17 | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | OR | OFF | 2005/06/24 15:56 |
| L19 | 1 | 1 and 18 | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | OR | OFF | 2005/06/24 15:56 |

| | | | | | | |
|---|---|---|---|---|---|---|
| L20 | 13079 | processing adj sequence | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | OR | OFF | 2005/06/24 15:56 |
| L21 | 0 | 18 and 20 | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | OR | OFF | 2005/06/24 15:56 |